tg-pubsub Documentation

Release 0.1.2

Thorgate

March 03, 2016

Contents

Contents:

CHAPTER 1

About

tg-pubsub provides easy pubsub for django models using redis messaging queue

1.1 Features

- Can mark own models as listenable
- Can mark external models as listenable
- Builtin pubsub server

CHAPTER 2

Installation

At the command line:

```
$ easy_install tg-pubsub
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv tg-pubsub
$ pip install tg-pubsub
```

Add to INSTALLED_APPS:

```
INSTALLED_APPS = (
    ...
    'tg_pubsub',
)
```

CHAPTER 3

Usage

Given a django model, for example:

```
from django.db import models

class Topping(models.Model):
    # ...
    pass
```

To make the `Topping` listenable it must use `ListenableModelMixin`:

```
from tg_pubsub.models import ListenableModelMixin

class Topping(ListenableModelMixin, models.Model):
    # ...
    pass
```

Now when a topping is created/updated a message is pushed to redis message queue on the following channels:

```
django
django:app_name-model_name
django:app_name-model_name:action
```

Start the pubsub server:

```
$ python manage.py pubsub_server
```

Use http://www.websocket.org/echo.html to connect to `localhost:8090` to see the messages being sent to the users

3.1 Limit instances

This can be controlled via the `should_notify(instance, action)` method on the listenable model class.

3.2 Permission checks

Every publish to the client will call the special `has_access(instance, user)` method on the listenable model class. Returning False means the user won't get a push for a model that they don't have access to.

3.3 Serialization

By default only the pk will be published by the pubsub server to the clients. To add more model fields one can use `serializer_fields` attribute on the listenable model. However, this will only work if the added fields can be serialized to json (by drf) and are available as instance attributes.

For more fine-grained serialization use `serializer_class` attribute and set it to a `django-rest-framework` serializer based on your needs.

3.4 Cant extend the model with ListenableModelMixin?

If you are unable to use `ListenableModelMixin` there is still hope. First you must declare your model listening config:

```
from tg_pubsub.models import ModelListenConfig


class HiddenModelListener(ModelListenConfig):
    model_path = 'some_app.Hidden'  # app_name.model_name of the model you want to listen to
```

And also add the path of your `ModelListenConfig` to `settings.TG_PUBSUB_EXTRA_MODELS`:

```
TG_PUBSUB_EXTRA_MODELS = [
    'path.to.HiddenModelListener',
]
```

ModelListenConfig supports the same features as `ListenableModelMixin`

CHAPTER 4

Settings

4.1 List of available settings

4.1.1 TG_PUBSUB_HOST

Network interface on which the pubsub server should bind to (default: `localhost`).

4.1.2 TG_PUBSUB_PORT

Port on which the pubsub server should listen on (default: `8090`).

4.1.3 TG_PUBSUB_PING_DELTA

Interval of periodical pings sent to clients, to disable pings set to False (default: `30`).

4.1.4 TG_PUBSUB_EXTRA_MODELS

List of import strings to subclasses of `ModelListenConfig` representing extra models that should be listened on (see: *Cant extend the model with ListenableModelMixin?*). (default: `[]`)

4.1.5 TG_PUBSUB_HELLO_PACKETS

List of import strings to callables that must return an instance of `BaseMessage`. These are sent to clients after successful connection has been established. (default: `[]`)

4.1.6 TG_PUBSUB_PROTOCOL_HANDLER

The protocol handler for your application (default: `tg_pubsub.protocol.RequestServerProtocol`).

Builtin protocols:

class `tg_pubsub.protocol.`**`RequestServerProtocol`**

WebSocketServerProtocol that gives handler a request-like object which might contain the session/user if token was provided.

class tg_pubsub.protocol.**SessionRequiredServerProtocol**
WebSocketServerProtocol which only allows handshakes with a valid token

class tg_pubsub.protocol.**AnyUserServerProtocol**
WebSocketServerProtocol implementation that allows any users (that provide a token)

class tg_pubsub.protocol.**AnonymousUserServerProtocol**
WebSocketServerProtocol implementation that only allows anonymous users

class tg_pubsub.protocol.**AuthenticatedUserServerProtocol**
WebSocketServerProtocol implementation that only allows authenticated users

class tg_pubsub.protocol.**StaffUserServerProtocol**
WebSocketServerProtocol implementation that only allows staff users

class tg_pubsub.protocol.**SuperUserServerProtocol**
WebSocketServerProtocol implementation that only allows superusers

Custom protocols should extend built-in protocols

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/thorgate/tg-pubsub/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" or "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

5.1.4 Write Documentation

tg-pubsub could always use more documentation, whether as part of the official tg-pubsub docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/thorgate/tg-pubsub/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *tg-pubsub* for local development.

1. Fork the *tg-pubsub* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/tg-pubsub.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv tg-pubsub
$ cd tg-pubsub/
$ python setup.py develop
$ pip install -r requirements_test.txt
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ make lint
$ make test
$ make test-all
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.

2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/thorgate/tg-pubsub/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_tg_pubsub
```

CHAPTER 6

Credits

6.1 Development Lead

- Jürno Ader <jyrno@thorgate.eu>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.1.2 (2016-03-03)

- Add support for automatic pings via (*TG_PUBSUB_PING_DELTA*)

7.2 0.1.1 (2016-01-22)

- Fixed some typos

7.3 0.1.0 (2016-01-21)

- First PyPI release

7.4 0.1.0-dev (2016-01-19)

- Initial implementation

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

A

R

S